SOFTWARE VALIDATION IN ISO17025: Dos, Don'ts and Difficulties

Author: Guy Snelling
Intercal PO Box 10907, Vorna Valley, 1686, South Africa
Email: guy@intercal.co.za
Phone: +27 11 315 4321 Fax: +27 86 515 2344

Abstract

The analysis by the Food and Drug Administration (FDA) of 3140 medical device recalls conducted between 1992 and 1998 revealed that 242 of them (7.7%) were attributable to software failures [1]. This is software written and installed by reputable firms that comply to FDA regulations. ISO17025 accredited laboratories often rely on software routines and Office scripts that are written in-house by a user that has a basic idea of how Excel works. How much more is this type of software likely to fail?
The reliance on our computers and software in the laboratory means that software validation must be a requirement to prevent errors occurring that, at best may be inconvenient and at worst, may prove fatal.
This paper will cover the types of software used in a laboratory, the general concepts and the importance of software validation, and will work through typical examples. This should enlighten the reader enough to be able to give software validation in the laboratory the attention that it needs and will point the way towards techniques that may be used.

1. Introduction

Testing and calibration laboratories that are accredited to ISO 17025 are required by the standard to validate many aspects of their management systems such as procedures, methods, and measurements to name just a few. One very important aspect that has often been glossed over under ISO17025:2005 however, is the validation of software. ISO17025:2017 is far more prescriptive regarding software and computer systems, although no methods for data management are given. This paper will highlight the general clauses of the standard, will cover the types of software used in a laboratory, the general concepts and the importance of software validation, and will give basic examples. These techniques can then be used a basis for even the most complex software.

2. Types of software

In general, there are five types of software used in a typical laboratory environment. Three of these fall under the title of 'commercial-off-the-shelf software', or COTS and would typically be software that is purchased from a reputable vendor. These include **operating systems** (OS) such as Windows, Linux, Apple OS and Unix, **productivity applications** like Microsoft's Office for general office work, MathWorks' Matlab for scientific and engineering applications, and Fluke's Metcal for calibration management. Laboratory information management systems (LIMS) from any of a number of suppliers also fall into this category. The third area of COTS would be embedded **firmware**, as used in electronic instrumentation, process and control system.

Usually it is not necessary to validate COTS as it is accepted that the vendor would have validated the software before release. However, even if this true for large vendors such as Microsoft, it can also be accepted that there is no such thing as bug-free software, even from the software giants. An example of this is Calculator, a program that ships with Microsoft

Windows. Calculator has had “The Perfect Square” bug in it since Windows XP. As an example, request the square root of 4, then subtract 2. The answer should be zero but, depending on the OS build and the PC system, the calculated and displayed result may be something like “-1.068281969439142e-19”. Almost zero, but not quite. Microsoft have recently fixed this “perfect square” bug in Windows 10, but if a vendor such a Microsoft produces software for 20 years with the same bug, then software from smaller vendors, and especially free-ware, share-ware and open-source software, should be treated with a large helping of suspicion.

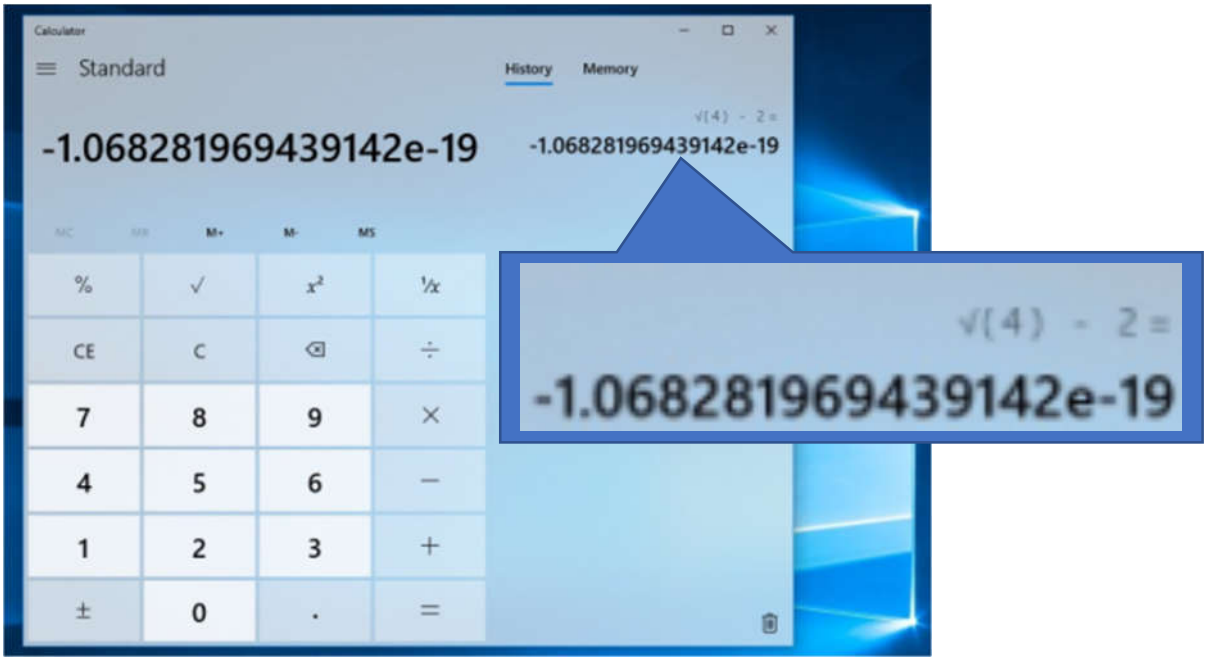


Figure 0 : An example of “The Perfect Square” bug in Windows Calculator

The fourth category of software most often found in a laboratory is ‘modified-off-the-shelf’ software, or MOTS. The obvious example of this is Microsoft’s Excel. While it may be accepted that the software itself performs correctly (as an example of COTS), it is the formulae, scripts and routines entered by the user that will require validation. We will be looking in detail at an example of this later on.

The final general category is custom written software. These may be written in any computer language such as Java and C++ and can also include more advanced programming of Microsoft Office products using VBA (Visual Basic). Like MOTS, no matter who writes this software, be it someone in-house or from an external software house, it must be validated.

Going back to embedded software briefly, firmware that is built into laboratory equipment does generally not need to be validated independently as the validation of this type of software would fall into the parameters of the validation or qualification of the equipment itself. It is important though to make sure the equipment qualification protocols will test all sections of the software. If firmware updates are then installed at any time, it is necessary to review the documentation from the vendor to see what changes have been made. If any updates include changes that would affect the results obtained from the equipment, then it may be required to re-validate the software or even to re-qualify the equipment. Because of this, it is obvious then that the version number of the firmware must be noted during equipment qualification.

3. Why Validate?

Truism: If builders built buildings the way programmers write programs, the first woodpecker to come along would destroy civilisation.

Why validate? To some, this might be an obvious question but to others it may not be as clear cut. The software used in the laboratory contributes to the quality of the results produced by the laboratory. It may be that the software is used to perform calculations on the raw measurement data or is used to control equipment in order to take those measurements. As with equipment qualification, software validation seeks to prove that the software is producing the results that it was designed for. It is also important to realise that it is not just regular calculations that have to be addressed during validation, but also the robustness of the software; that is the software's ability to handle unexpected inputs.

In July 2016 British chain store Marks & Spencer were embarrassed when they had to withdraw their quarterly report a few hours after releasing to the stock market. Incorrect data inserted into a spreadsheet showed a 1.3% increase in sales instead of a 0.4% drop. The embarrassment was compounded by the fact that a cursory calculation of nearby figures would have shown the error. In other words, the spreadsheet had not been validated.

In another example, retail giant Convivality inadvertently exaggerated its profits by £5 million due to a spreadsheet error. This contributed to the empire losing £500 million in 3 weeks as its shares plummeted.

In both cases it would seem that simple input checking and validation of formulae would have prevented these situations from occurring.

4. What should be validated?

ISO17025:2017 regards software as equipment and groups it along with measuring instruments, standards and reference materials:

> **6.4 Equipment**
> **6.4.1** The laboratory shall have access to equipment (including, but not limited to, measuring instruments, software, measurement standards, reference materials, reference data, reagents, consumables or auxiliary apparatus) that is required for the correct performance of laboratory activities and that can influence the results.

This shows that software must be treated and validated with at least the same level of importance as the validation of all other equipment used in the laboratory.

As with all validations in the laboratory, there must be a documented validation procedure for each piece of software to be validated. For a validation to give legitimate results, it is important that every aspect of the software be addressed during the validation. An understanding of how the software operates is thus vital to be able to produce a suitable validation procedure, and it follows then that good software documentation is needed, especially for MOTS and custom software. This documentation will form the basis of the validation and should therefore give a

good description of the workings of each section of the software and its expected output. As pointed out previously, change-notices from the vendor are also important documents.

Obviously, the validation must check that the correct outputs are generated by the software for a given input, but it must also look at how unexpected input values are handled. It may be that a measurement procedure will need to be modified to include a note to the operator about specific values or messages returned by the software and these should be addressed during validation.

An example of this may be how a humidity data logger handles a measured value that is greater than 100 %rh. If the logger output is shown as "over" or "+++" for example, then there is no problem but if the output value is always "99.9 %rh" for any measured value above this then any software that uses this as an input must determine if the measured value is really 99.9 %rh or is in fact larger. Similarly, a temperature data logger with a faulty probe may give a high temperature value rather than an error message. This would need to be addressed in the software and this functionality must be suitably validated.

True Value	Logger 1	Logger 2
95	95	95
96	96	96
96	96	96
97	97	97
97	97	97
98	98	98
99	99	99
99	99	99
100	Err	99
100	Err	99
101	Err	99
102	Err	99
103	Err	99
104	Err	99
106	Err	99
104	Err	99
106	Err	99
108	Err	99
107	Err	99
105	Err	99
102	Err	99
100	Err	99
99	99	99
98	98	98
97	97	97
97	97	97
96	96	96
96	96	96
95	95	95

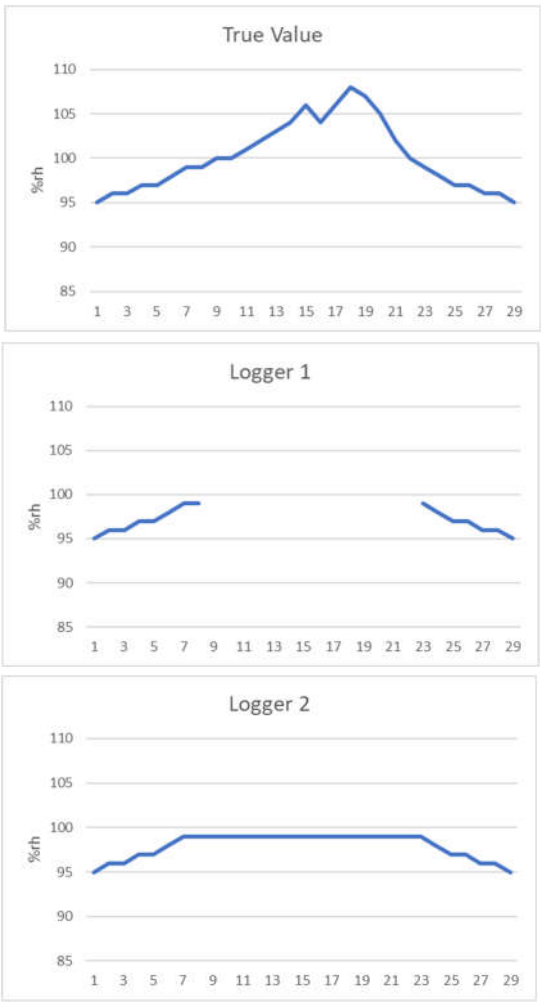


Figure 1 : An example of over-range handling in humidity loggers

5. ISO17025:2017

The standard concerns itself with the "Control of data and information management" in section 7.11 but as we have seen in section 6.4, software is considered to be equipment in the laboratory and must be treated the same as hardware and relevant paragraphs in section 6 therefore also apply.

> **7.11.2** The laboratory information management system(s) used for the collection, processing, recording, reporting, storage or retrieval of data **shall be validated for functionality**, including the proper functioning of interfaces within the laboratory information management system(s) by the laboratory **before introduction**. Whenever there are any changes, including laboratory software configuration or modifications to commercial off-the-shelf software, they shall be authorized, documented and **validated before implementation**.
>
> **6.4.4** The laboratory shall verify that equipment **conforms** to specified requirements **before** being placed or returned into service.

From paragraphs 7.11.2 and 6.4.4 we can see that it is a requirement to validate software and changes (including updates) before they are implemented.

It is also worth mentioning NOTE 2 of 7.11.2:

> **NOTE 2** Commercial off-the-shelf software in general use within its designed application range can be considered to be sufficiently validated.

However, it should again be pointed out that no software is bug-free and the validation of COTS should still at least be considered.
Truism: There are two ways to write error-free programs; only the third one works.

> **7.11.3** The laboratory information management system(s) shall:
> a) be protected from unauthorized access;
> b) be safeguarded against tampering and loss;
> c) be operated in an environment that complies with provider or laboratory specifications or, in the case of non-computerized systems, provides conditions which safeguard the accuracy of manual recording and transcription;
> d) be maintained in a manner that ensures the integrity of the data and information;
> e) include recording system failures and the appropriate immediate and corrective actions

Sub-paragraphs 7.11.3 a) and b) tell us that all software must be protected against tampering and loss. This means that cells in Excel must be locked and worksheets must be password protected.
The reference to "loss" here may also refer to protecting client data from malicious damage or theft, possibly by malware or by hacking.
Sub-paragraph 7.11.3 d) means that once data has been entered and accepted, by the issuing of a calibration certificate for example, the raw data as entered by the user must be protected from change. It also means that an adequate backup procedure must be in place.

Sub-paragraph 7.11.3 c) refers to operating in a suitable environment, air-conditioned server room for example. Remembering that software is considered to be equipment in this standard, the requirement for a suitable environment is reinforced earlier in section 6:

> **6.3.3** The laboratory shall monitor, control and record environmental conditions in accordance with relevant specifications, methods or procedures or where they influence the validity of the results.

> **7.11.4** When a laboratory information management system is managed and maintained off-site or through an external provider, the laboratory shall ensure that the provider or operator of the system complies with all applicable requirements of this document.

In paragraph 7.11.4 we see that even if the laboratory doesn't write any of their own software, be it custom software or MOTS, it is still the responsibility of the laboratory to ensure that the software is validated.

> **7.11.5** The laboratory shall ensure that instructions, manuals and reference data relevant to the laboratory information management system(s) are made readily available to personnel.

Paragraph 7.11.5 means that the software must be well documented and that instructions for its use are readily available to laboratory staff. This can easily be achieved in Excel for example by the use of "notes" in individual cells.

> **7.11.6** Calculations and data transfers shall be checked in an appropriate and systematic manner

"Data transfer" referred to in paragraph 7.11.6 may be the manual entry of a displayed value into an Excel worksheet. It may therefore necessary to include data validation routines into specific cells in the worksheet to ensure that valid data has been entered. Calculations in cells should be performed by checks against manual calculations performed during the software validation phase.

6 Risk Analysis

ISO17025:2017 does not have a lot to say with regard to how laboratories are to meet the objectives of the standard, such as with validating software. Instead it leaves the responsibility with developing methods to the laboratory. What this means is that the laboratory must analyse the risk involved in its activities and must take appropriate action.

General risk analysis falls outside of the subject of this paper, but we can take a look at it with regards to software validation.

Examples of questions to be asked when analysing the risk for software would be "What would be the consequence if this calculation is incorrect?" or "What would be the consequence if this software does not perform correctly?" The answer to these questions will determine the level of validation that will be required. For example, the validation of a word processor used to type procedures would require limited validation, if any. Software that calculates or controls

the amount of reagent used in a PCR analysis, on the other hand, would require stringent validation.

7 How to validate

Truism: Testing can only prove the presence of bugs, not their absence.

> **6.4.3** The laboratory shall have a procedure for handling, transport, storage, use and planned maintenance of equipment in order to ensure proper functioning and to prevent contamination or deterioration.

Paragraph 6.4.3 refers to the need for a documented procedure for the planned maintenance of equipment in order to prevent contamination or deterioration. In terms of software, deterioration and contamination can refer to file corruption or even manual tinkering by a user. While validation is required before putting software into use, it is also a requirement that the current version of software be validated regularly, in accordance with a procedure to check for any form of corruption. Using a documented procedure ensures that the validation covers all the required aspects of the software and that the validation is performed in a similar manner each time.

7.1 General Methods

There are two broad methods to validate software, namely "black box" and "white box" testing.

"Black box" testing ignores the underlying software and simply requires analysis of the actual output against an expected output, with a given series of inputs. The software should have been written to calculate values as required by a laboratory procedure or method and the validation test values must be based on this document.

7.2 Examples in Excel Worksheets

Figure 1 shows an example from Excel where the required output is the sum of the square roots of the input values. Note that the SUM function has also been manually validated.

	A	B
1	Input	Output
2	16	4
3	13	3.605551
4	Sum	7.605551

Validation Test Data:

Input A2 : 16 Expected Output B2 : 4 Found: 4 ✓

Input A3 : 13 Expected Output A3 : 3.605551 Found: 3.605551 ✓

Expected Sum: 7.605551

Found: 4 + 3.605551 = 7.605551 ✓

Figure 2 : An example of "Black Box" testing an Excel worksheet

"White box" testing requires access to the underlying code or formula so that it can be validated against the requirement of the procedure or method as shown in figure 3.

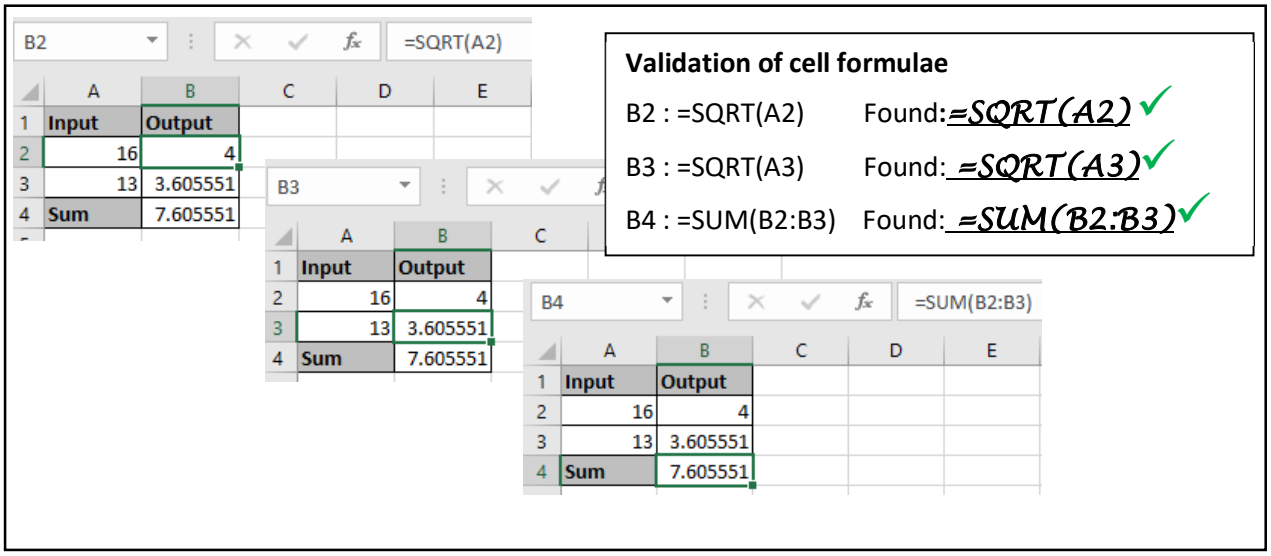


Figure 3 : An example of "White Box" testing an Excel worksheet

The two methods are complementary and a combination of them can be used, depending on the complexity of the software and the requirement of the procedure against which the validation is to be performed.

Error handling can also be validated by either or both of the methods as shown in the examples in figures 4 and 5.

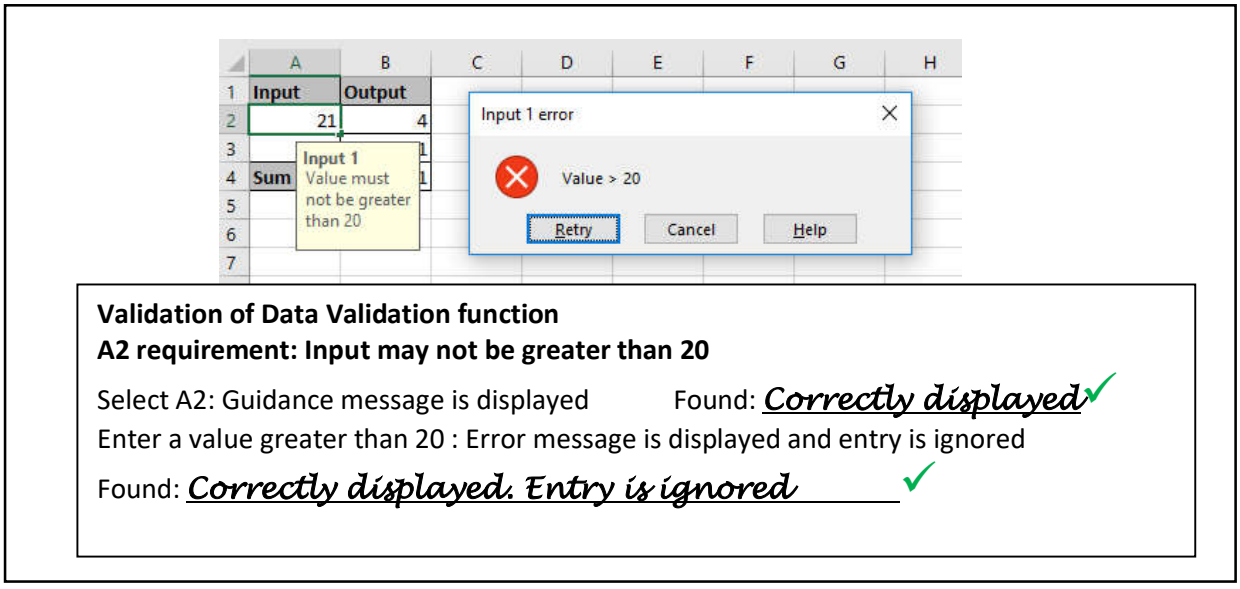
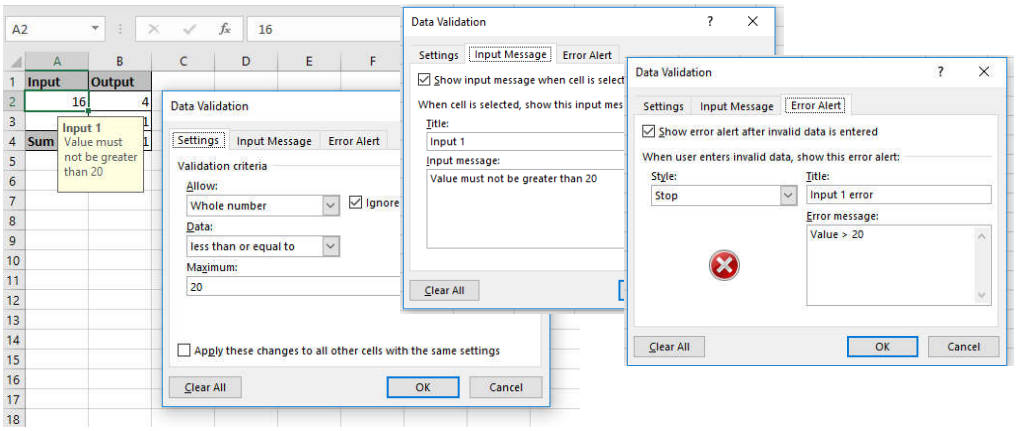


Figure 4 : Black Box testing of error handling in Excel



Validation of Data Validation function
A2 requirement: Input may not be greater than 20

Settings: Allow whole number less than or equal to 20 Found: *Correct* ✓

Input message: Value must not be greater than 20 Found: *Correct* ✓

Error alert: [STOP] Value > 20 Found: *Correct* ✓

Figure 5 : White Box testing of error handling in Excel

7.3 Examples in Code

While the formulae entered into Excel cells can be classified as "one-line" code, most software uses multiple lines of code. Examples that may be typically found in a laboratory are in Microsoft Office macros written in Visual Basic (VBA). Figure 6 shows a small part of such a macro used by this author to read information from data loggers files and to insert this data into an Excel worksheet. The snippet opens a text file containing comma separated values from the datalogger. It then selects the correct column of data and pastes it into the working excel worksheet. It then closes the text file.

```
'Get the relevant data from the selected logger
‘Select the correct column
Windows(logfile(I) + ".txt").Activate
Select Case T
   Case 1
      Range("E7").Select
   Case 2
      Range("G7").Select
   Case 3
      Range("F7").Select
End Select

‘Copy to clipboard
 Range(Selection, Selection.End(xlDown)).Select
 Selection.Copy

 'switch to main file and paste
 Windows("LOGGERDATA.xls").Activate
 ActiveCell.Offset(0, 1).Select
 ActiveSheet.Paste

 'close the file
 Windows(logfile(I) + ".txt").Activate
 Application.CutCopyMode = False
 ActiveWorkbook.Close False
```

Figure 6 : Snippet of VBA code from an Excel macro

Such a routine can be either Black Box or White Box tested.

Black Box testing could use the routine to process a sample text file and then compare the final worksheet with a master test file. If they compare then the routine has not been corrupted and can be considered valid.

White Box testing could be accomplished by using a program to compare each line of the routine with a master program to see if any changes have been made.

7.4 Checksum

One simple method of White Box testing is to use a checksum, by whichever method is convenient to the laboratory. A check sum of the working version of the routine is generated and is the compared to the checksum of the master file. If they are equal then the working routine is valid and may be used. This an extremely simple and rapid method of validating code and may therefore be used before every instance that the working routine is to be used.

The checksum method cannot stand on its own however as it is unable to show where any corruption in a file may have occurred if the checksum test fails.

```
7f17e8e971d1f517419d876e04eae7a8
```

Figure 7 : The MD5 HASH checksum of the code in Figure 6

7.5 Security

Security of the software should also be checked to ensure that unauthorised changes cannot be made. One of the biggest causes of errors in spreadsheets is allowing users access to key control functions. Cells containing formulae should be protected and this protection must be checked as part of the validation procedure. In Black Box testing the user should attempt to select cells that are protected. In White Box testing the protection flag of the cell and the protection status of the worksheet should be checked.

Note that the worksheet should be protected by a strong password and the control of this password must be documented. This will ensure that the worksheet can be accessed only by persons authorised to do so by the laboratory quality system.

8. Documenting the validation

There are several methods of documenting the validation of software and these include the following:

8.1 Combined procedure and report

The procedure for the validation can include checklists and areas for comment at the relevant steps. Once completed and signed, the document forms the final report and can be filed for reference. The procedure should include space to enter the name or filename of the software, the revision number, the date and the name and signature of the person performing the validation. This ensures traceability of the version being validated. Figure 8 shows how such a document could be formatted for the example in Figure 1.

SOFTWARE VALIDATION PROCEDURE

Filename: Example 1 **Revision:** 1.0

1. Test Data

Make the following entries:

Cell	Input Value
A2	16
B2	13

2 Results

Compare the expected values with the displayed values:

Cell	Expected Value	Displayed Value
A3	4	*4*
B3	3.605551	*3.605551*
B4	7.605551	*7.605551*

3 Conclusion

The software is validated and is suitable for use.

Date: *4 May 2018* Validated by: *A.N. Other*

Figure 8 : Possible validation procedure and report for the example in Figure 2

8.2 Separate procedure and report.

The procedure can be written with a report section at the end that can be printed off and completed and filed separately. There must then be traceability between the procedure and the report so both should carry the software name or filename and the revision number.

8.3 Report in the software

In the case of Excel files it is possible to validate in the worksheet itself. An extra worksheet is added that will contain the validation criteria and space for relevant entries. Using the example from figure 1, figure 9 demonstrates the principle.

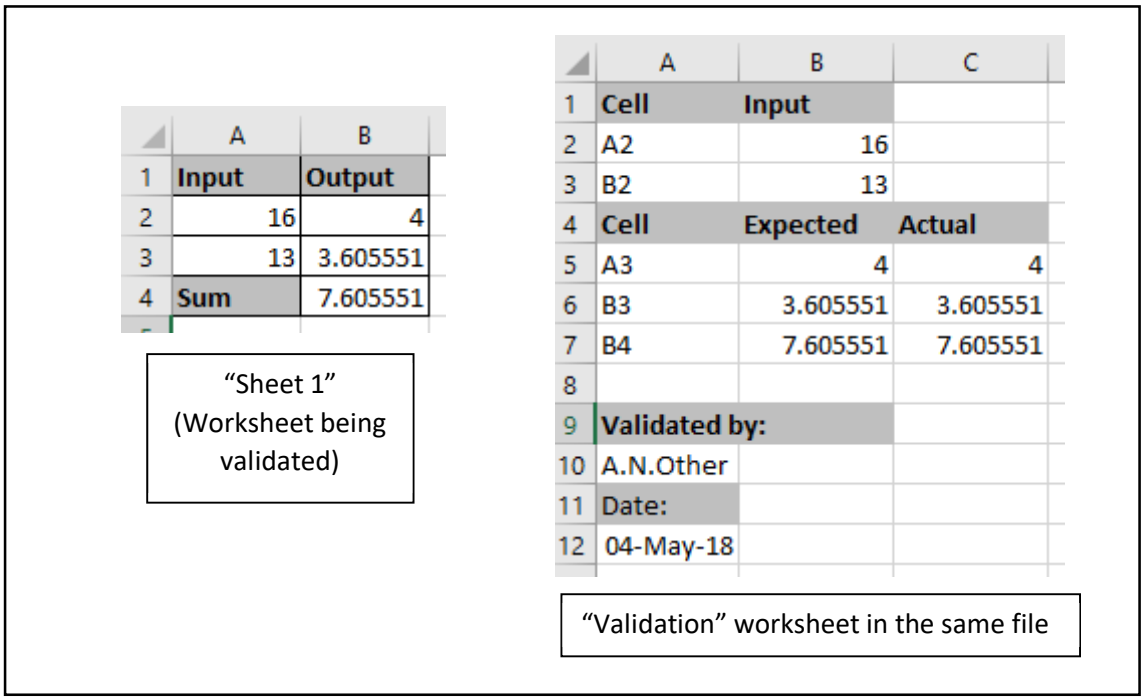


Figure 9 : Example of validating directly in an Excel file worksheet

Once validation is completed the entire file is saved separately as proof of validation and a blank copy is kept as a working master.

9 Conclusion

In a world that relies so heavily on computers, it is important to ensure that the programs running on these computers, process information correctly and give consistent and reliable results. Performed correctly, controlled and documented software validations are vitally important to the day-to-day operations of ISO17025 accredited facilities.

References

International Software Testing Qualifications Board (ISTQB)
http://istqbexamcertification.com/why-is-testing-necessary/ – retrieved 05 April 2018

General Principles of Software Validation; Final Guidance for Industry and FDA Staff
https://www.fda.gov/downloads/MedicalDevices/.../ucm085371.pdf – retrieved 05 April 2018

http://www.softwaretestinghelp.com/iq-oq-pq-software-validation/ – retrieved 05 April 2018

https://www.fda.gov/downloads/RegulatoryInformation/Guidances/ucm125125.pdf – retrieved 05 April 2018

https://mybroadband.co.za/news/technology/254933-microsoft-fixes-decade-old-bug-in-calculator.html – retrieved 05 April 2018

http://news.softpedia.com/news/microsoft-fixes-decade-old-windows-calculator-square-root-bug-520559.shtml#sgal_0 – retrieved 05 April 2018

https://www.thetimes.co.uk/article/spreadsheets-plus-human-error-can-add-up-to-disaster-sknj23b2g - retrieved July 2018